

# IT Deployment RunBook



**Audience:** IT administrators and DevOps engineers deploying the product in enterprise environments (Windows Server, on-premises, or private cloud). This page is an operational workflow — install, configure, monitor, update, secure. For deep product reference, each section links to the corresponding technical page.

## What You'll Deploy

A typical enterprise deployment has three layers:

- **Runtime server** — one Windows Server machine that runs the solution (tags, alarms, historian, scripts) as a Windows Service. This is the core of any deployment.
- **Client stations** — operator workstations running the Rich Client (WPF), or users accessing the solution through web browsers (HTML5 /WebAssembly).
- **Integration points** — PLCs, databases (SQL Server, Oracle, PostgreSQL), historian storage, and optionally MQTT brokers, OPC UA servers, or cloud endpoints.

For architecture context and sizing, see [Platform Architecture Reference](#) and [Deployment Scenarios](#).

Other deployment modes (Linux, Docker/Kubernetes, edge devices, mobile clients) are supported but outside the scope of this runbook. See the links in section 10 below.

## 1. Plan

Before installing, decide:

- **Runtime placement** — dedicated server (recommended) or shared application server. The runtime is a single executable plus supporting services.
- **Service account** — a domain service account (preferred) or Local System. Domain accounts allow auditing and controlled access to network resources, databases, and file shares.
- **Historian database** — built-in SQLite (small deployments), external SQL Server / PostgreSQL / Oracle (typical), or time-series databases (InfluxDB, TimescaleDB).
- **Ports and firewall** — see the Network Ports section below. Open only what you need, inbound only where required.
- **License model** — per-server licensing with online or offline activation. See [Licensing Procedures Reference](#).
- **High availability** — optional runtime redundancy (primary/secondary). See [Redundancy Reference](#).

## 2. Install

### Prerequisites

- Windows Server 2019 or later (Windows 10 / 11 for small deployments)
- .NET Framework 4.8 (included in Windows 10+)
- .NET 8 runtime (required for cross-platform runtime, AI integration, and DataHub)
- Visual C++ 2015-2022 Redistributable (bundled with the installer)
- Administrator rights on the target machine

### Interactive install

Run the signed installer and follow the wizard. Default install location is `C:\Program Files\Tatsoft\FrameworX\`. Full step-by-step reference: [Installation and Licensing Reference](#).

### Silent / unattended install

The installer is an Inno Setup package and supports standard silent switches for SCCM, Intune, PowerShell DSC, Ansible, or any deployment pipeline:

```
Start-Process -FilePath "FrameworX-10.1.5-Setup.exe" `
-ArgumentList "/VERYSILENT", "/SUPPRESSMSGBOXES", "/NORESTART", `
"/LOG=C:\Logs\frameworx-install.log", `
"/DIR=C:\Program Files\Tatsoft\FrameworX" `
-Wait -PassThru
```

**Common switches:**

Switch	Purpose
/VERYSILENT	Fully silent — no progress, no prompts

/SUPPRESSMSGBOXES	Suppress all message boxes (use with /VERYSILENT)
/NORESTART	Prevent automatic reboot on completion
/DIR=" <path> "	Override default install directory
/LOG=" <path> "	Write install log to specified file

#### Install exit codes:

Code	Meaning
0	Success
1	Setup failed to initialize or user aborted
2	User cancelled during preparation
5	Install aborted (user cancelled during copy)
3010	Success — reboot required to complete

### File-copy install (.NET 8 runtime only)

The .NET 8 runtime can also be deployed by copying the installation folder. Useful for portable deployments, custom update pipelines, and locked-down environments where MSI-style installs are blocked. See [Runtime Installation Reference](#).

## 3. Configure

### Install folders

Standard install layout:

Location	Purpose
C:\Program Files\Tatsoft\FrameworkX\	Executables and supporting libraries (read-only after install)
C:\Users\Public\Documents\FrameworkX\	User-facing content: solutions, exchange folder, screenshots, trace logs (default)
C:\ProgramData\Tatsoft\FrameworkX\	Optional location for operational data when configured for enterprise deployments

Full layout reference: [Installation Folders and Utilities Reference](#).

### Run the runtime as a Windows Service

For production, the runtime should run as a Windows Service so it starts automatically with the machine, survives user logouts, and is managed through standard Windows tooling (`sc.exe`, `Services.msc`, `Get-Service`).

Service install options:

- **Designer UI** — Runtime Startup has an AutoStartup button that installs the runtime as a Windows Service
- **Command-line** — use `TManageServices.exe` (located in the product root folder) from an elevated shell

Example: install the runtime as a Windows Service for a specific solution, unattended:

```
cd "C:\Program Files\Tatsoft\FrameworkX"
.\TManageServices.exe /installtstartup `
  /solution:"C:\Users\Public\Documents\FrameworkX\Solutions\Plant1.tproj" `
  /silent

# Uninstall later
.\TManageServices.exe /uninstalltstartup /solution:"Plant1" /silent
```

`TManageServices.exe` also handles `TWebServices`, `TSecureGateway`, `TMQTTBroker`, `THardkey`, and `RuntimeMCPHttp`. Run `TManageServices.exe /?` for the full command list. The `/silent` flag makes it script-friendly: it never pauses for input and returns a non-zero exit code on failure. Full reference: [Runtime Startup Reference](#).

Run the service under a dedicated domain account where possible. The account needs:

- Log on as a service
- Read access to the solution folder
- Write access to log and data directories
- Network access to PLCs, databases, MQTT brokers, and OPC UA servers as required by the solution

## Web server configuration

Web client, REST endpoints, and runtime HTTP services are hosted by the built-in web server. Configure ports, TLS certificates, and optional reverse-proxy routing as described in [Web Server Configuration Reference](#).

## Licensing

Online activation is used for internet-connected servers. Offline activation (license file transfer) is supported for air-gapped networks. See [Licensing Procedures Reference](#).

## Network ports

Typical enterprise deployment ports:

Port	Protocol	Service	Direction
3101	TCP	Runtime client-server	Inbound from Rich Clients
3110	HTTP	Web client (non-TLS)	Inbound from browsers
3111	HTTPS	Web client (TLS)	Inbound from browsers
4840	TCP	OPC UA Server (optional)	Inbound from OPC clients
1883 / 8883	TCP / TLS	MQTT broker (optional)	Inbound from MQTT clients
1433 / 5432 / 1521	TCP	SQL Server / PostgreSQL / Oracle	Outbound from runtime to DB

Default runtime and web ports are configurable. Exact values depend on solution configuration — see [Runtime Startup Reference](#) and [Server Configuration Reference](#).

## 4. Monitor

### Log locations

The runtime writes trace logs by default to `C:\Users\Public\Documents\FrameworX\TraceLogs\`. The location is configurable per solution via Solution Settings; enterprise deployments typically relocate it under `ProgramData` so it's picked up by Windows-standard log collectors (Splunk forwarder, Datadog agent, Azure Monitor, ELK).

Install logs are written to the path specified by `/LOG=` during silent install, or to `%TEMP%\Setup Log YYYY-MM-DD.txt` for interactive installs.

### Service lifecycle events

Windows logs basic service start, stop, and unexpected termination events for every installed FrameworX Windows service to the **System** event log under source `Service Control Manager`. SCOM, Azure Monitor, Datadog, PRTG, and other Windows-native monitoring tools pick these up automatically without any custom configuration — point your alerts at the service name (for example, `TStartup-<SolutionName>`).

Application-level events (solution load failures, license errors, module exceptions) are written to the runtime TraceLogs. When an Event Log entry indicates a service failed to start, the root cause detail is in the most recent TraceLog file.

## Runtime System Monitor

The built-in System Monitor exposes live metrics on tags, modules, memory, and client sessions. See [Runtime System Monitor Reference](#) and [Runtime Diagnostics Reference](#).

## 5. Update & Rollback

Updates within the same major version (e.g. 10.1.5 to 10.1.6) can be applied by running the newer installer over an existing installation. The installer supports side-by-side versioning with rotation, so a minor update can be rolled back by re-running the previous installer. Full semantics: [Managing Updates and Multiple Versions Reference](#).

Before updating a production server:

- Back up the solution folder (`C:\Users\Public\Documents\FrameworX\Solutions\`)
- Back up the historian database (if external)
- Stop the runtime service gracefully
- Run the installer with silent switches
- Start the service and verify via System Monitor and the Windows System event log

## 6. Uninstall

Uninstall from Programs & Features or silently:

```
"C:\Program Files\Tatsoft\FrameworX\unins000.exe" /VERYSILENT /SUPPRESSMSGBOXES /NORESTART
```

The uninstaller removes the installed executables, the Windows Service registration, and the registry entries. User content under `C:\Users\Public\Documents\FrameworX\` is preserved — remove manually if you want a fully clean uninstall.

To remove **only** the installed Windows services (without removing the product files), use:

```
cd "C:\Program Files\Tatsoft\FrameworX"  
.\TManageServices.exe /removeallservices /silent
```

## 7. Secure

Production deployments should follow the hardening checklist. Topics include TLS configuration, authentication (local users, Windows authentication, Active Directory), role-based access control, secrets storage, audit logging, and network segmentation.

- [Security Hardening Reference](#) — detailed procedures
- [Security Hardening Compliance How-to](#) — step-by-step hardening guide
- [IEC 62443 Compliance How-to](#) — industrial cybersecurity standard
- [NERC-CIP Compliance How-to](#) — critical infrastructure
- [FDA 21 CFR 11 Compliance How-to](#) — regulated industries

## 8. Automate

Typical automation patterns:

- **SCCM / Intune** — package the silent installer as an application deployment with the switches shown in section 2. Use exit code 3010 as a soft-reboot signal.
- **PowerShell DSC / Ansible / Chef** — wrap `Start-Process` with the silent switches and check exit codes for the product installer; use `TManageServices.exe /silent` for service install/uninstall steps.
- **Solution deployment** — copy the `.dbsln` solution file to the target server's Solutions folder, then run `TManageServices.exe /installstartup /solution:"<path>" /silent` to install the runtime as a service. Solutions are self-contained.
- **Bulk uninstall** — call `unins000.exe` with silent switches across a fleet, or `TManageServices.exe /removeallservices /silent` to clean services only.

## 9. Troubleshoot

Symptom	First place to look
Service won't start	Windows <b>System</b> event log (source <code>Service Control Manager</code> ) for the service start failure; then open the most recent file in the <code>TraceLogs</code> folder for application-level detail
Port conflict	<code>netstat -ano</code> to find what's on the port; change port in <code>Runtime Startup</code> config
License error	Runtime <code>TraceLogs</code> for the detailed license error, then <a href="#">Licensing Procedures Reference</a>
Client can't connect to runtime	Firewall rule on runtime port, <code>Test-NetConnection</code> from client, runtime client-server logs
Web client blank / auth error	TLS certificate validity, browser console, <a href="#">Web Server Configuration Reference</a>
Database connection fails	Service account permissions, connection string, firewall rule to DB server

Deep reference: [Runtime Troubleshooting Reference](#).

When opening a support ticket, collect: the most recent `TraceLog` file, the install log, the output of `Get-Service` for the runtime service, the relevant Windows System event log entries for the service, and the solution build number.

## 10. Other Deployment Modes

This runbook focuses on typical enterprise Windows Server deployments. Other modes are supported — see the dedicated references:

- [Container Deployment Reference](#) — Docker and Kubernetes
- [Linux Deployment Reference](#) — .NET 8 on Linux servers and edge devices
- [Redundancy Reference](#) — primary/secondary runtime pairs
- [SecureGateway Services Reference](#) — multi-site communication routing
- [Mobile App](#) — mobile client deployment

## 11. Remote management via SolutionCenter API

From 10.1.5 GA, the recommended pattern for managing many FrameworkX installations from a central place is the **SolutionCenter API** — a JWT-authenticated REST surface that every installation exposes on port 10108 (TWebServices). Central tools call it directly; bulk fan-out happens at the orchestration layer (Ansible, PowerShell DSC, custom dashboards) rather than per-machine WinRM or shipped agents. Surface covers solution lifecycle (list/run/stop), file ops, license activation, machine settings, and runtime connection inspection.

Activation-gated **default-OFF** in 10.1.5 — operators must explicitly enable per installation; see [SolutionCenter API Reference](#) for the activation procedure, scope model, endpoint catalog, and OpenAPI 3.1 document URL.

The product-install-time and service-lifecycle patterns from sections **2** and **8** above (silent installer, TManageServices.exe, SCCM / Intune / PowerShell DSC) are still the right tools for the install + service-registration layer of a fleet rollout. The SolutionCenter API operates one layer above — once installations exist, it manages what runs on them.